AD_____

Award Number: W81XWH-04-2-0010

TITLE: Supporting the Virtual Soldier with a Physics-Based Software Architecture

PRINCIPAL INVESTIGATOR: Thomas J. Impelluso, Ph.D.

CONTRACTING ORGANIZATION: San Diego State University
San Diego, CA 92182-1931

REPORT DATE: June 2005

TYPE OF REPORT: Final

PREPARED FOR: U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-06-2005	Final	1 Jan 2004 - 31 May 2005

4. TITLE AND SUBTITLE
Supporting the Virtual Soldier with a Physics-Based Software Architecture

5a. CONTRACT NUMBER

5b. GRANT NUMBER
W81XWH-04-2-0010

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Thomas J. Impelluso, Ph.D.

E-Mail: impellus@kahuna.sdsu.edu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
San Diego State University
San Diego, CA 92182-1931

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This effort researched a new methodology to integrate the modules of mechanics to support digital organ geometries of the virtual soldier. Specifically, this effort researched a computational methodology to solve for the problems of flexible multi-body dynamics. This methodology can readily be extended to multi-phase and multi-scale problems in general to support organ geometries. Rather than presenting a new theory, a new algorithm or a new software package, this effort introduces a methodology so that researchers in mechanics can deploy their preferred algorithms or theorems in a new way to solve such problems by building up a preferred solution method from the fundamental technologies of the cyber-infrastructure. The approach advocated in this phase 1 testbed is: fault tolerant, extensible and scaleable.

15. SUBJECT TERMS
Finite element, multi-body dynamics, multi-phase computations

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
U	U	U	UU	34	

Table of Contents

INTRODUCTION

This effort researched a new approach to solving coupled problems (multi-phase and multi-scale) problems in mechanics.

At this time, many research groups are solving multi-phase (e.g.: solid/fluid interaction, flexible multi-body dynamics) and multi-scale (continuum mechanics coupled with molecular dynamics) problems using large physics packages (legacy or re-constructed), which are then integrated using large network packages (e.g.: Globus). While this is in keeping with traditional academic approaches of building upon existing solutions, it is inimical to the light-weight, extensible, scaleable and fault-tolerant coding that is required for supporting organ geometries with physics.

This effort researched a simpler approach, and one which places value on the years of efforts of the computational mechanicians who have developed the fundamental algorithms of mechanics.

This effort builds a multi-phase solution method – deformable (finite element) links (multi-body dynamics), by integrating the algorithms from the bottom up.

Rather than create a large-scale system, this research creates the integration by building upon the fundamental network inter-process communication protocols: sockets, fork/exec, shared memory and so on.

1. Such an approach is scaleable: different modules can be run on targeted hardware;
2. Such an approach is extensible: more phases and scales can be added (even to a solution that has already commenced);
3. Such an approach is fault tolerant: if one module (software or hardware) goes down, the system keeps running.

4. BODY OF REPORT

The Body of this Report consists of, in the following order:

1) Most recent quarterly report - together to give a history of the approach taken: the mistakes and successes as the various technologies were tested

   THIS IS FOUND IN PAGES 3-7 inclusive

2) A research paper that is now in submission to the Journal of Computational Mechanics – this constitutes the bulk of the report as it details the design of the experimental platform.

   THIS IS FOUND IN PAGES 8-26 inclusive.

QUARTERLY REPORT FORMAT

1. Award No. W81XWH – 04 – 2 – 0010
2. Report Date: January 10, 2005
3. Reporting period: October 1, 2004 – December 31, 2004
4. Principal Investigator: Thomas J. Impelluso
5. Telephone No.: (619) 594-0753
6. Award Organization: DARPA
7. Project Title: Physics Based Software Architecture
8. Current staff, role and percent effort of each on project.

STAFF MEMBER	Role	% EFFORT
Thomas Impelluso	PI	25%
Richard Harris	Researcher	50%
Angel Perez	Researcher	100%

9. Contract expenditures to date (as applicable):

COST ELEMENTS	THIS QUARTER	CUMULATIVE
Personnel	7,040.00	64,384.25
Fringe Benefits	1,302.63	5,742.60
Supplies		
Equipment		
Travel	242.56	2.208.64
Other Direct Costs	10,000.00	13,834.00
Subtotal	18,585.19	86,169.49
Indirect Costs	9,519.34	45,339.00
Fee		
Total	28,104.53	131,508.49

10. Comments on administrative and logistical matters. None
11. Use additional page(s), as necessary, to describe scientific progress for the quarter in terms of the tasks or objectives listed in the statement of work for this contract. Explain deviations where this isn't possible. Include data where possible.

Summative Statement of Accomplishments: Q1 and Q2 and Q3

In keeping with the ongoing scheme, Q1 and Q2 are re-reported and are followed by Q3.

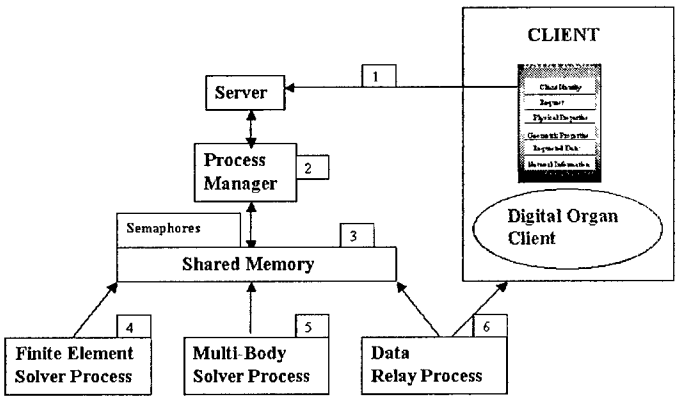
Q1 EFFORT

1) Q1. Creation of optimized finite element (FE) code for deformation of objects.
   a. Input and Output files are identical
   b. Code written as an equilibrium state machine

2) Q1. Creation of optimized multi-body dynamics (MD) code for motion of linked systems.
   a. Input and Output files are identical
   b. Code written as a equilibrium state machine

3) Q1. First pass at Integration architecture. Schematic below

***SHARED* MEMORY ARCHITECTURE**



Here FE, MB, and Relay are processes on the same machine.

Sequence of Operation

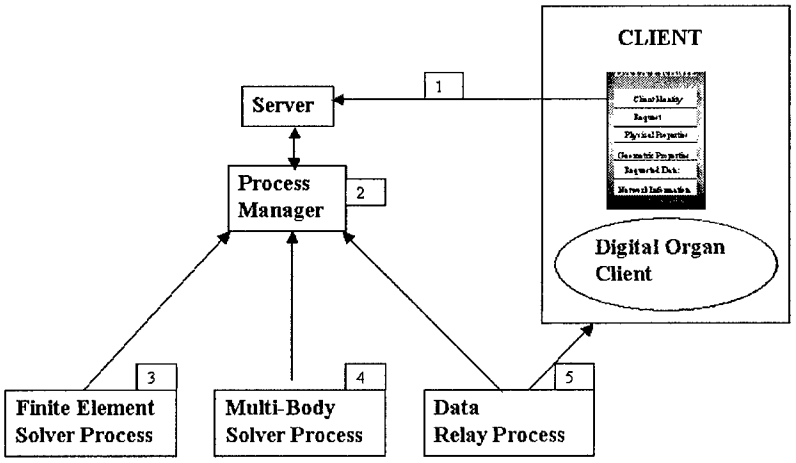
1. Client connects and requests physics using defined data frame (blue schematic).
2. Server forks process manager to handle request
3. Process manager creates: shared memory and semaphores
4. Process manager forks FE code
5. Process manager forks MB code
6. Process manager forks Reader to relay data back to client

This concept was tested and abandoned, as it would prove futile for more than two or three simple physics processes.

Q2 EFFORT

4) Q2. Second pass at Integration architecture **completed**. Schematic below.

***DISTRIBUTED* MEMORY ARCHITECTURE**



Here, each process, FE, MB, and Relay is on a different machine.

Sequence of Operation

1. Client connects and requests physics using a data frame (blue schematic)
2. Server forks process manager to handle request
3. Process manager forks FE code
4. Process manager forks MB code
5. Process manager forks Reader to relay data back to client

This scenario is scaleable and extensible and does not preclude shared memory as a subset. However, the control exerted by the process manager is still too loose and it would be difficult to absorb legacy software.

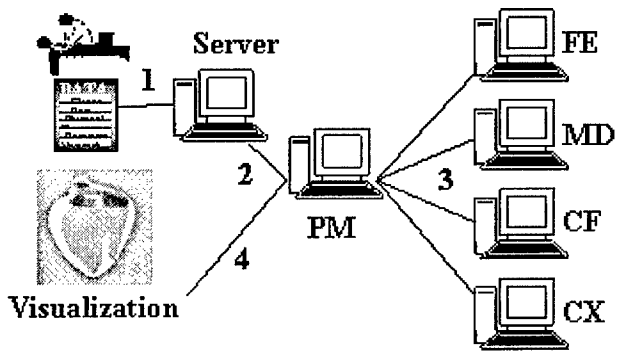
Q3 EFFORT

5) Q3. Third pass at Integration architecture **completed.** Schematic below

It was soon evident that if this platform is to incorporate a diverse array of processes, an entirely new approach was needed. The new processes would encompass: a) the algorithms of mechanics: finite element, fluids, dynamics, contact; b) the algorithms of cellular effects: Brownian motion, Molecular dynamics; c) legacy software of preexisting platforms: existing heart code, HIP code, etc.

This new approach is more lightweight, distributed, and extensible. It allows for fault-tolerance.

***DISTRIBUTED* MEMORY**
ARCHITECTURE WITH INVERSION OF CLIENT/SERVER



Sequence of Operation

1. Client requests assistance from server using defined data frame (blue schematic)
2. Server forks a process manager: PM
3. PM will "ssh" (a secured shell remote fork) the appropriate processes on targeted hardware.
   - 3a. These processes – Finite Element (FE), Multi-Body Dynamics (MD), Computational fluid mechanics (CF) and Contact (CX) – are activated.
   - 3b. The aforementioned processes open as servers
   - 3c. The PM connects as a client to all (inverting the client/server paradigm)
4. The PM brokers the solution and delivers the data back to a visualization client.

Goals for Q4

1) Make the architecture hardware independent and OS independent.
2) Remove the LabView initial client request and replace with generalized data frame format protocols.
3) Clean software from dead code.
4) Writing of papers, presentation at conferences, crediting to DARPA VS program.
5) Begin designing data frames

6) Begin designing better GUI's

Long Term Goals beyond Phase 1

Begin to test deployment of

1. Generalize mechanics server to spawn: FE, MD, Fluids Contact
2. Generalized cellular codes
3. Generalized legacy codes

Q4 EFFORT

1. No new revisions of architecture.
2. Fault tolerance development commenced.
3. Optimization of all software commenced.
4. First presentation of work at Medicine Meets Virtual Reality 13 (see attachment for award)

A Proposed Cyber-Infrastructure to Solve Multi-Phase Problems in Mechanics

Impelluso, T.

Department of Mechanical Engineering, San Diego State University, San Diego, CA 92182

Abstract: This paper introduces a new methodology to solve for the problems of flexible multi-body dynamics. However, this methodology can readily be extended to multi-phase and multi-scale problems. Rather than presenting a new theory, a new algorithm or a new software package, it introduces a methodology so that researchers in mechanics can deploy preferred algorithms or theorems in a new way using fundamental technologies of the cyber-infrastructure. This approach is intrinsically collaborative in nature and returns a voice to the computational mechanician without access to large scale resources. The US NSF intends to spend one billion dollars per year funding research associated with the cyber-infrastructure. At this time, it is the computer scientist who is leading the charge toward the solution of coupled systems, expecting that mechanicians exploit existing communication packages.

Keywords: Client Server, Visualization, Distributed Memory, Virtual, Finite Element, Multi-Body Dynamics, Multi-Phase Problems, Cyber Infrastructure.

1. INTRODUCTION

This paper introduces a new paradigm to solve for the problem of flexible multi-body dynamics. This paper does not purport to establish new algorithms or theoretical models; nor is this paper concerned with the solution of a specific problem of interest in multi-body dynamics, for its own sake. Rather, the research presented in this paper shines a light on a new direction for researchers in the field of computational mechanics. This paper presents an approach to create new types of solution methods that can be readily extended to encompass problems such as solid/fluid interaction, flexible multi-bodies in contact with fluid, or problems that require the bridging of the length scale gap (multi-phase problems).

The cyber-infrastructure" [Atkins, 2004] is used to designate the plethora of technologies, from low level inter-process communication facilities, to higher level packages. The NSF intends to fund one billion dollars per year in this area1 and this paper introduces new cyber-infrastucure research vistas within the context of flexible multi-bodies. This new method can also be used to integrate the following modules of mechanics: finite element methods, computational fluid mechanics, and multi-body dynamics.

Some might argue that there are network packages, developed by computer scientists, which are capable of integrating large-scale physics codes. However, using those packages places mechanicians in a position subservient to the computer scientists who developed them. While, ostensibly, there is nothing wrong with this, this paper argues that the fundamental technologies are simple and that mechanical engineers could readily embrace and exploit them. With the results presented here – by an example in flexible multi-body dynamcis – mechanicians can expand their work into disciplines such as surgical planning, physics based virtual reality or physics based simulations for entertainment, education and training – work that is now being conducted by computer scientists in ways that lack the rigor that mechanicians can bring. This paper does not introduce a methodology in competition with others researchers in the discipline of flexible multi-bodies (whose work is far superior to the simple approach taken here). Rather, this paper demonstrates how existing solution schemes can rapidly expand; it embraces all theoretical solution methods to the problem of flexible multi-bodies, and then some.

---

1 The author of this paper was an invited participant at the June 6-7 NSF panel on the CI.

2. BACKGROUND

Before proceeding with the description of this new approach, certain definitions are provided. Some might feel argue that a description of higher level packages that could be used to integrate the various modules of mechanics are in order in lieu of the more fundamental ones presented here. The author responds by saying: 1) the technologies presented herein are core to all higher level packages anyway; 2) by learning these technologies, computational mechanicians can work as peers with computer scientists; 3) these technologies are simple and can be readily implemented into ongoing multi-phase solution techniques.

All the function calls listed below, in **`bold courier`**, are simply function calls that any computer program can make, as seamlessly as they make IMSL function calls.

2.1 Computer Science

2.1.1 Process: A *program* is a file containing an instruction set, while a *process* is the running instantiation of the instruction set. A user can execute a *program* an arbitrary number of times, and each time creates a unique *process.* The user owns the processes. There is a special type of process, a *daemon,* which is not owned by a user, but is owned and managed by the operating system.

2.1.2 Sockets: A socket is the standard underlying software interface for all computer network communications. Sockets operate with four basic steps whose functionality can best be described with a "telephone" analogy. First, a process creates the socket with use of the **socket** function; this is analogous to installing a telephone jack in the wall. Next, the process invokes the **bind** function that is analogous to associating a number with the telephone. The invocation of the **listen** function is analogous to notifying the telephone company that the telephone is ready to receive calls. At this point the process issues the **accept** function which is analogous to picking up the telephone once it rings. Meanwhile, a remote process invokes two functions – **socket** and **connect** – and then the two processes, can **`read`** and **`write`** to each other as easily as the read and write from and to files on disk.

2.1.3 Fork/Exec: A **fork** is a system function that enables a process to copy itself: to reproduce itself. Both processes, the original and the copy, continue execution from the point of the fork. Fork technology also enables a process to determine if it is the original or the copy. When a process makes an **exec** function call, it stops what it was doing and replaces itself with a new process; basically, the **exec** function instantiates an entirely new

program. Used in conjunction with a **fork**, this is a powerful mechanism allowing one process to instantiate another while maintaining the integrity of peripheral resources and shared access to data stack and heap. For example, after a process **forks** a child process, the child identifies itself and then **execs** a different process in its place. In this way, one parent process can control other processes.

2.1.4 Client/server model: A server process is structured so that it runs on a certain computer, as a *daemon*, DEFINE!! and waits for service requests from client processes running on other computers. The clients communicate with the server through a socket, and request functionality. Upon receipt of a request from a client the server will fork a copy of itself. The child copy then execs the requested functionality, while the original parent returns to receiving new requests from other clients. (Incidentally, any computer, regardless of its power, can become a server simply by being the machine on which a process opens a networks socket to service clients.)

2.1.5 `ssh`: Secure Shell (SSH) is a technology that enables one process to log onto a remote computer over a network to execute commands – processes – on that remote computer. User intervention is not required. When a process makes an **`ssh`** call, it provides the name of the targeted machine, and the process it wishes to execute. SSH functionality is layered on top of the standard socket libraries.

2.2 Mechanics

The field of *mechanics* represents the study of the behavior of the physical world and is divided into three major areas: theoretical mechanics, computational mechanics and applied mechanics, as indicated in Figure 1. Theoretical mechanics focuses on the interactions of objects. Computational mechanics represents the use of the computers and algorithms to solve the problems of mechanics. Applied mechanics focuses on real world engineering and scientific applications such as machine design or systems analyses.

The sub-discipline of computational mechanics can be further subdivided into three focus areas: nano-mechanics, continuum mechanics and multi-body dynamics. Nano-mechanics is concerned with particle response at an atomic level and involves Brownian motion as well as molecular dynamics. Continuum mechanics is concerned with the behavior of bodies at a macroscopic level for which the microstructure is homogenized by phenomenological averages. Multi-body dynamics is concerned with mechanical assemblies and is the foundation of robotics and auto mechanics. Of these three, continuum mechanics can be further subdivided into yet three more

areas: the study of solids, fluids, and multi-physics (the latter including solid/fluid interaction problems).

2.2.1 Multi-Body Dynamics (MD):

Consider a multi-body system consisting of N_R rigid bodies. The position of the j^{th} rigid body at each time t is described by the position vector of the center of mass $\mathbf{r}_j$ and the orthogonal transformation matrix $\mathbf{A}_j$ whose columns are the unit base vector of the local body-fixed coordinate system. A material point of the j^{th} rigid body is described by using the local coordinate system $\mathbf{s}'_j$ (which remains constant):

$$\mathbf{r}_j(t)=\mathbf{A}_j(t)\,\mathbf{s}'_j, \quad j=1,2,\cdots,N_R \ . \tag{1}$$

As the position of the j^{th} rigid body changes, both $\mathbf{r}_j$ and $\mathbf{A}_j$ must be found by integrating the velocity of the mass center $\dot{\mathbf{r}}_j \equiv d\,\mathbf{r}_j/dt$ and $\dot{\mathbf{A}}_j \equiv d\mathbf{A}_j/dt$. The time derivative of $\mathbf{A}_j$ can be described by the angular velocity $\boldsymbol{\omega}'_j$ expressed in skew-symmetric matrix $\tilde{\boldsymbol{\omega}}'_j$ as follows (Nikravesh, 1988; Haug, 1989):

$$\dot{\mathbf{A}}_j=\mathbf{A}_j\,\tilde{\boldsymbol{\omega}}_j', \tag{2}$$

where the prime denotes that primed components are with respect to the body-fixed local frame.
It is noted that by virtue of Euler's theorem, the orthogonal rotation matrix $\mathbf{A}_j$ is also expressed by the rotation vector $\boldsymbol{\pi}'_j$ in terms of the Euler angles (or parameters).

A multi-body system consists of rigid bodies connected at joints. Each connection is described as a constraint. For example, if body j and body k are connected at joint P, the connection is described by

$$\mathbf{r}_j+\mathbf{A}_j\mathbf{s}_j^{P\prime}=\mathbf{r}_k+\mathbf{A}_k\mathbf{s}_k^{P\prime}, \tag{3}$$

where $\mathbf{s}_j^{P\prime}$ denotes the position of joint P with respect to the j^{th} local frame. In addition, the system may be driven by a prescribed motion of some joints. All of these may yield N_C constraint equations:

$$\Phi_\alpha(\mathbf{r}_1,\cdots,\boldsymbol{\pi}_1,\cdots,t)=0, \quad \alpha=1,2,\cdots,N_C \tag{4}$$

Each constraint can also be imposed by utilizing the Lagrange multiplier: λ_α. Lagrange's equation of motion for body j is expressed with respect to the virtual displacement of the mass center $\delta \mathbf{r}_j$ and the virtual rotation of the body-fixed frame $\delta \boldsymbol{\pi}_j$ as

$$\delta \mathbf{r}_j^T (m_j \ddot{\mathbf{r}}_j - \mathbf{F}_j - \mathbf{F}_j^C) + \delta \boldsymbol{\pi}_j'^T (\mathbf{J}_j' \dot{\boldsymbol{\omega}}_j' + \tilde{\boldsymbol{\omega}}_j' \mathbf{J}_j' \boldsymbol{\omega}_j' - \mathbf{n}_j' - \mathbf{n}_j^{C\prime}) = 0, \tag{5}$$

where m_j, $\mathbf{F}_j$, $\mathbf{J}_j'$, and $\mathbf{n}_j'$ are, respectively, the mass, the external force, the moment of inertia matrix, and the external moment acting on body j. In (5) the superscript 'T' accompanying a vector denotes the transposition of the vector. The constraint force and moment are defined as

$$\mathbf{F}_j^C = \sum_\alpha \lambda_\alpha \frac{\partial \Phi_\alpha}{\partial \mathbf{r}_j}, \qquad n_j^C = \sum_\alpha \lambda_\alpha \frac{\partial \Phi_\alpha}{\partial \boldsymbol{\pi}_j}. \tag{6}$$

For some connections that are permanent, one may use the joint position to describe the center of mass to reduce the total number of equations.

The time integration of the Euler-Lagrange equations of (5) is often performed by either using the Adams-Bashforth and Adams-Moulton predictor-corrector method (Haug, 1989) or the Runge-Kutta method (Nikravesh, 1988).

2.2.2 The Finite Element (FE) Method:

The deformation of a continuum occupying the region Ω_0 at time t=0 is described by a differentiable deformation map □ between the current position **x** in Ω_t at time t and the position **X** at time t = 0:

$$\mathbf{x} = \phi(\mathbf{X}, t). \tag{7}$$

The formulation in Eq. 7 is referred to as the Lagrangian (material) formulation.

Let □, □, **b**, $\mathbf{v} \equiv \partial \mathbf{x} / \partial t$, and Dv /Dt denote the Cauchy stress tensor, mass density, body force per unit mass, velocity vector, and acceleration vector, respectively. The deforming continuum satisfies Cauchy's equations of motion (8), the rate constitutive relations (9), and the flow rule for the irreversible (plastic) part of the rate of

deformation tensor $\mathbf{d}_p$, and the boundary conditions on velocity **v**, and traction **h** on $\partial\Omega_t$ (the boundary of Ω_t) (10).

$$\nabla\cdot\sigma+\rho\mathbf{b}=\rho\frac{D\mathbf{v}}{Dt} \text{ in } \Omega_t, \tag{8}$$

$$\frac{D\sigma}{Dt}-\omega\cdot\sigma-\sigma\cdot\omega^T=\mathbf{C}_{el}:(\mathbf{d}-\mathbf{d}_p) \text{ in } \Omega_t, \tag{9}$$

$$\mathbf{v} \text{ or } \mathbf{h}=\sigma\cdot\mathbf{n} \quad \text{specified on } \partial\Omega_t. \tag{10}$$

where D( )/Dt denotes the material time derivative (by keeping **X** fixed), $\mathbf{C}_{el}$ is the elastic modulus tensor, □ $\equiv\nabla\times\mathbf{v}$ (= curl **v**), and $\mathbf{d}\equiv(\nabla\mathbf{v}+\mathbf{v}\nabla)/2$ is the rate of deformation tensor.

In the FE formulation, the deformation map is interpolated at each nodal point, *i.e.*, at node k, $\mathbf{x}_k=\phi(\mathbf{X}_k,t)$. The weighted residual form (11) of (8) and (10) is integrated over each finite element with respect to the virtual velocity □**v** that vanishes on $\partial\Omega_t$ where **v** is prescribed:

$$<\delta\mathbf{d},\sigma>_\Omega=<\delta\mathbf{v},\rho(\mathbf{b}-\frac{D\mathbf{v}}{Dt})>_\Omega+<\delta\mathbf{v},\mathbf{h}>_{\partial\Omega}, \tag{11}$$

where $<,>_\Omega$ and $<,>_{\partial\Omega}$ denote the inner product over Ω_t and $\partial\Omega_t$, respectively.

In most FE computations for solids, the Lagrangian formulation, **x**(**X**, t) and **v** (**X**, t), is employed where the FE mesh moves with the material points. Therefore, the computation of the acceleration term Dv/Dt simply becomes the partial derivative with time $\partial\mathbf{v}/\partial t$.

3. COUPLED PROBLEMS AND DISTRIBUTED COMPUTING

3.1 Multi-Phase Problems

The analysis of coupled mechanical systems was developed in the 1970s by three groups: Northwestern University, California Institute of Technology (CIT), and Lockheed Palo Alto Research Labs (by J. A. DeRuntz, C. A. Felippa, T. L. Geers and K. C. Park). This initial work focused on different applications and evolved into different problem-decomposition techniques. For example, Belytchko and Mullen at Northwestern considered node-

by-node partitions whereas Hughes and Liu at CIT developed what eventually became known as element-by-element methods. The Lockheed group began its investigation in the underwater-shock problem for the Office of Naval Research in 1973. In this work an FE computational model of the submarine structure was coupled to Geers' "doubly asymptotic" boundary-element model of the exterior acoustic fluid, which functions as a silent boundary. In 1976 Park developed a *staggered solution procedure* to treat this coupling. A more detailed historical overview as well as a summary of the methodology is given in the aforementioned work by Belytchko and Mullen.

The formulation of the equations of motion of a flexible multi-body system invites coupling in many ways. Geradin and Cardona (Geradin, 2000) formulated an approach in which the inertial frame becomes the reference frame. Coupling of FE and MD has also been done with linear (Goncalves, 2001) and non-linear (Ambrosio, 1992) FE methods. Contact has also been introduced using unilateral constraints (Pfeiffer, 1996) or continuous contact forces (Lankarani, 1994). The availability of state variables in a multibodies allows for different control paradigms in the framework of vehicle dynamics, biomechanics or robotics (Valasek, 1999) The coupling of fluid and structural dynamics allows for solid fluid interactions in which there is large rotations of system components (Moller, 2000).

3.2 Distributed Computing:

Distributed computing began in the early 1970's with the arrival of minicomputers and ubiquitous networks. While the smaller computers were less powerful than their larger mainframe counterparts, they were also much cheaper. An easy way of scaling a computation was to buy more of such machines and distribute a problem over the multiple CPUs. Distributed computing differs from other parallel computation models in that the CPU nodes are autonomous and that all connections and procedure calls are accomplished by message passing or socket communication.

Software for managing such memory communication can either be constructed from simple sockets, commercially purchased or found within the open source community. One of the more powerful commercial packages is Constellation (http://www.rti.com) which is based on robot controller modularity. The open source solutions vary from very thin wrappers built on top of TCP/IP and UDP/IP, through RPC (Remote Procedure Call) based solutions and on up to MPI (Message Passing Interface) and MPI2 solutions (http://wwwunix.mcs.anl.gov/mpi/index.html) and the Parallel Virtual Machine

(http://www.csm.ornl.gov/pvm/pvm_home.html). The goal of these distributed computing packages remains focused on communication, coordination, guaranteed reliability, transaction processing and synchronization. Some of the systems proceed further into memory management, implementation of virtual shared memory protocols, multi-language and platform functionality, latency management and many other enhancements to improve the performance of the distributed application.

One of the larger and more interesting open source projects is the Globus Alliance (http://www.globus.org). This is a research and development project focused on enabling the application of Grid concepts to scientific and engineering computing. Typical applications include: "smart instruments", teraflop desktops, collaborative engineering, and distributed supercomputing. The system consists of packages such as: resource allocation managers, authentication services, network analysis monitoring systems, and secondary storage systems. Globus enables researchers to share large scale software systems across a "global" network. Such large scale software systems include heart models, physiology models, weather models, manufacturing models and many others.

3.3 Limitations

The methods discussed in Section 3.2 do not utilize the methods described in Section 3.1. The methods of 3.2 are not scaleable. They cannot be extended to encompass new physics modules. Once they are running, new physics modules cannot join process groups, nor can physics processes leave groups once they are no longer needed. The systems are not fault tolerant: certain physics algorithms cannot be restarted with new parameters without having to restart the entire system. Processes cannot be easily targeted for certain types of platforms. The results cannot be easily delivered to clients that might need them. It is still difficult to port data from one process group to another.

Some dedicated platforms such as heart and lung models have evolved into massive systems, and packages like Globus will connect them at the top and transport data between them. However, in such legacy systems, the FE code is less a general FE code and more specific to, say, heart mechanics.

This proposed effort integrates the algorithms of mechanics and builds to a larger platform from the bottom up by assembling modules that have no "embedded knowledge" of the type of organ or system being analyzed. This approach proposes to return to fundamental algorithms of mechanics and connect them with simple inter-process communication facilities by exploiting the CI technologies presented in Section 2.1. As one begins to solve more

complex coupled problems, it is necessary to keep the data communication facilities under tight control and to keep the mechanics algorithms separate from each other. In this way one can build to any type of model with basic building blocks of both mechanics and the CI, while keeping the physics processes and their numerical issues separate from each other and separate from the more complex communication facilities.

4.0 A NEW APPROACH TO SOLVING COUPLED PROBLEMS

This paper now introduces three concepts critical to solve coupled problems using the cyber-infrastructure: 1) a new philosophy of what constitutes a physics code; 2) the nature of data frame formats for transmission of physics data; 3) a proposed client/server design (Fig. 2).

4.1 A new view of physics coding.

First, consider the spirit of the UNIX/Linux operating system, wherein an operating system command is a data converter (e.g., an operating system command can covert a .doc file into a .pdf file). A physics code should operate on a data set the same way. Rather than viewing a physics code – or process – as large scale packages equipped with solution environments for pre and post processing (ADAMS, NASTRAN come to mind), one could instead view a physics code as an operating system command that converts data.

The author suggests that it is time to consider whether the format of the input file for a finite element code should be identical to the format for an output file. It would be a mistake to assume this is a trivial matter, for it requires the understanding the FE solvers become iterators that enforce equilibrium. In this way, the same FE process could read both input and output files. As a result of adhering to such formatting rules the FE code can also be viewed as a data converter: a user should be able to take an output file, modify a stress, and feed it back into the same FE code. This promotes a view of physics coding wherein an iterative FE code is not just an analytical tool, but also an equilibrium enforcer on a data set. Essentially, the FE process becomes one which simply ensures a minimum residual stress after balancing the internal and external virtual work.

Similarly, a multi-body dynamics (MD) process could also be conceived as one which reads a configuration and performs one time step increment of the Runge-Kutta methods (which are typical in such multi-body problems). In fact, whether it be a computational fluids, molecular dynamics, finite element methods and so on, a physics process should read a data set, enforce equilibrium on that data sent and output that same data set in the same format as the input file.

Physics processes could very well be operating system commands. To carry out this work the author created an FE and MD program according to the format laid out here, and registered them as operating system processes. In this case, the FE code could read a data set, decide if it has rights to act on it, and if it does, inspect the data set to ensure equilibrium and finally output the equilibrium state in the same format.

4.2 A Data Frame for transmission of mechanics data.

Next, consider the nature of a data frame that describes a particular problem, but from the historical vantage point of the accomplishments of the field of seismology – which, incidentally, constitutes an entire division at NSF and which is already active in NSF's new cyber-infrastructure initiatives.

The Standard for the Exchange of Earthquake Data (SEED) is an international standard format for the exchange of digital seismological data. SEED was designed for use by the earthquake research community, primarily for the exchange between institutions, of unprocessed Earth motion data. The SEED database defined the protocols by which seismic data was transferred over the Internet. In so doing, it enabled researchers to develop hardware and software tools to facilitate seismic research and analysis.

In a similar way, a second aspect of the work presented here is to demonstrate the proper design of such protocols that enable algorithms to communicate with remote physics servers. For example, data frames structures for *transmission* and data process structures for *analysis* should be identical to each other as this would facilitate memory acquisition for processing and communication between physics processes. The type of process (FE or MD for example) will reside in the header (initial bytes) of the data frame. Other pertinent information will also reside there.

A Typical Data Frame is presented in Figure 1. I is designed to include all pertinent information including (in an order that will be the subject of research): material properties, geometric properties, boundary conditions, numerical methods constants, target architecture, and backup architecture. Naturally, physics processes will communicate with each other and this communication will adhere to rigorous data frame designs. Researching data frame formats for the transmission of physics-based simulations is a critical task if one hopes to deliver the power of mechanics to real-time simulators.

In fact, the file formats for input and output – already expected to be identical - should also be identical to the data frame format that ports physics data over the internet. In this way, a physics process, on receipt of a data frame from a remote computer, inspects the header, determines if it has rights to act on that data, enforces equilibrium upon it, if it does, and then finally exports the data back out to the internet. At this point one only then needs an efficient client/server design for physics processing. It is this preliminary design that was developed in the research presented here.

4.3 The Client/Server system for coupled problems

The entire architecture is now presented. There are seven codes that will now be discussed. They are summarized in Table 1. Following this table the codes are referenced by their "designation" (third column) in the order of their functionality.

Name	Purpose	Designation	Parent
Control Client	Requests a Solution	CC	User
User Server	Starts the system	US	User
Known Server	A Daemon Server to run indefinitely	KS	US
Process Manager	Manages the solution of a specific problem	PM	KS
Finite Element	Conducts deformation analysis	FE	PM
Multi-Body Dynamics	Conducts dynamics analysis	MD	PM
Visualization	Inspects results	VX	User

The system must be started. A user executes the User Server (US) process via a normal command line execution on a machine that will host the server. The first thing this process does is to detach itself from the terminal (or user) to become a daemon process. This is done via the standard **fork/exec** described earlier. The child of the US is a daemon server known now as the Known Server (KS) (while DS might appear to be a more suitable designation, the parlance of client/server technology suggests that KS – Known Server – is more appropriate) which can fulfill requests but is not associated with a user or a terminal. The KS runs on the same machine as the US. The US, however, terminates and is never heard from again. This might seem a trivial operation. However if a long-term goal is to create a client/server system that solves for multi-scale and multi-phase problems, then to prepare for such extensibility, this sort of rigor is required.

The KS then issues the standard calls to open a socket: `socket`, `bind`, `listen`, `accept`. Again, these

are trivial calls codable by a mechanican to develop a template architecture2. The KS hangs (blocks) on the **accept**: it waits for connections from a client requesting a solution.

Meanwhile, the Control Client (CC) program is activated by the user and becomes a process. The CC process issues the standard **socket**and **connect** calls to initiate communication with the host machine on which the KS is running. It is the CC that is used to set up the problem. Basically, this machine is to be considered the pre-processor that defines the problem and requires a solution

Upon receipt of a connection from a CC, the KS now spawns a process manager (PM) to manage this particular multi-phase problem. This is done via an **ssh** call to a remote machine on its database list of potential assistants. Using **ssh** functionality, the KS informs the new remote PM of the identity of the remote CC who is requesting the solution. Furthermore, the KS also informs the CC of the identity of the machine on which the PM is running: basically, it introduces the CC (analyst) to the PM (solver). At this point, the KS can step out of the picture and service new requests should they arise. Henceforth, the KS is no longer be referenced in this discussion.

The role of the PM is to orchestrate the solution in an organized fashion. The first thing the PM does is to issue the standard socket calls to wait for several connections – the first of which comes from the CC. The CC connects to the PM as a client (in fact, the first socket connection that the PM allows is the one from the CC). The PM and that CC are now talking using standard **read/write**.

The PM then must now wait for instructions from the CC. The CC informs the PM of the specific problem: an FE and an MD code are needed initially to solve for a flexible multi-body system. Upon receipt of a data frame describing the problem, the PM must now execute as many FE codes as there are meshes, and the requested MD code. This is described next.

Exactly how the PM instantiated each physics process became an issue in this research due to how **ssh** tunnels operate between processes. For example, one option is to allow the PM to execute the FE and MD codes on remote machines via an **ssh** call. However, as a general rule, operating system complexities can pass through this **ssh** tunnel and this needed to be avoided. To accomplish this, the PM first **fork**'ed itself for each physics process that was needed. Then, each of these child PM's inititaes the **ssh** of the remote physics process, and informs that

2 Fault tolerance and such otehr issues require furthr coding by a comp. sci person could then be brought in. TALK

physics process of the identity of the parent PM. Each child PM then terminates and the instantiated remote physics process connect back to the PM via a socket.

Before continuing, another issue to emphasize is that if there are several meshes in the system, each mesh would require its own unique FE process. Thus, for the deformable multi-body system such as the femur, tibia and foot, three FE processes and one MD process was instantiated. The reason for this will be addressed shortly, but for now, they will be known as FE1, FE2, FE3 and MD, or, alternatively, as FE^i and MD.

Continuing, the remote physics processes each create a socket through which they connect back to the PM as clients of the PM. The final schematics is shown in Figures x. The first thing each physics process – each FE^i and MD – does is to pass back to the PM a header data frame that identifies the nature of the physics code to the PM (is this an FE^i or an MD). The PM then delivers the appropriate data to each FE^i and MD code (this includes mesh geometry, initial and boundary conditions and other material properties). The PM "knows" how to solve this problem (via the design of the computational mechanician). Each physics process - FE^i and MD - then calculates the size of its solution and then informs the PM of the size of the data that it will be working on and returning to the PM. The PM can now orchestrate the solutoin.

In this first effort presented here, the MD process starts the solution. The PM seeds the MD code (and at this time the FE^i all wait). The MD performs a step in the Runge-Kutta, Newton-Raphson solution and calculates link positions, velocities, accelerations, body forces and contact. MD delivers this back to the PM which then strips out what is needed for each FE^iE process and sends that data on to them. Each FE^i can now run an analysis on each independent mesh since it now knows the associated body force, and boundary conditions. Each FEi solves its designated problem as a simple FE code and delivers stress and position to PM. But PM (in this case does not deliver such to MD as it is not needed. The PM holds the data in the event that a visualization client connects and desires to view the solution.

Next, the authors address the reason for executing an independent FE process for each mesh. At this time, contact is an algorithm within a finite element code. The authors suggest that contact should become its own independent process – CX. In this way, the FE code retains its integrity as a simple solver (equilibrium enforcer) for the non-linear equations that define deformation theory, and there can be a process for each independent mesh. Each

FE[1] process can deliver geometrical information – via the PM – to the CX process. The CX process then inspects each mesh to determine if there is penetration within a mesh or between meshes. The CX process then delivers nodal resistance forces to each mesh that needs them, via the PM. In this way, CX can be viewed as an equilibrium establisher on mesh data sets, maintaining the new view of physics coding. Furthermore, the input/output design of the physics codes themselves facilitates this view.

Of course, at any time, the visualization client (VZ), can enter and request data. The request comes via a socket connection to the PM that then periodically interrupts the solution to deliver the data that is requested. A multitude of VZ clients can request data and they need not request the same data.

Finally, the authors now address validation, but briefly. For a three member revolute linkage system subjected to the force of gravity, the results of material stresses were compared with that predicted by the commercial package Marc. Results compared very well but the authors defer on a rigorous validation study at this time since more work needs to be done. This approach is in its infancy and any coupled system analyzed this way would be better analyzed with a commercial package. For the goal here is simply to put forth the required technologies in an attempt to demonstrate to computational mechanicians an alternative method to solve coupled problems using the technologies of the cyber-infrastructure.

5.0 DISCUSSION

As an illustrative example, consider the musculoskeletal system from a macroscopic perspective. This system consists of linkages (bones) whose movements are induced by forces from actuators (muscles). Muscles, and to a lesser degree, bones, are deformable objects, which are in contact with each other. Blood flows through veins and arteries in this system. An accurate phenomenological analysis of this system requires the algorithms of multi-body dynamics (MD) for the motion of the bones, finite element methods (FE) for the deformation of the muscle and bone, computational fluid mechanics (CF) to model blood flow, and contact algorithms (CX) to model contact.

In this a fully developed platform, CI technologies would be used to create a platform to analyze the aforementioned coupled system. The various physics processes – FE, MD, CF, and CX – would remain independent, general and simple. A control process would execute the aforementioned physics processes and regulate their activity. It would request information from one physics process and deliver it to another. It would monitor numerical convergence and stability, and restart each singular process if necessary, without restarting the

entire system. This approach advances the state of the art in the application of advanced information technology to coupled problems in science and engineering.

In this system, processes can come and go. A solution can be started, and, much later, the analyst could request a new phase be added. For example, a flexible multi-body with contact may not require a fluids process at all. It might be only later in the solution method that such information is required, and once it is obtained the analyst could always, kill the fluids process, or even the finite element process for flexibility. As mechanicians we must ask ourselves why we are building such large scale legacy systems when the preferred approach would be to keep solution methods light weight. Large scale, legacy systems – often commercial – are inimical to the freedom of research that a system presented herein presents. In such a system, power is returned to the computational mechanician focused on certain modules.

Finally, one could add more processes to this mix: heat transfer (HT) and re-mesh (RM), the latter to re-mesh an FE domain. In addition, the server could fork the following PM's: mechanics-PM, microscopic-PM, biology-PM. Of these, the mechanics-PM, in turn, forks the FE, MD, CF, CX, HT and RM processes; the microscopic-PM forks molecular dynamics and Brownian motion; the biological-PM forks bioinformatics processes and so on. Assuming a particular problem manifests a length scale gap in the ontology one need only define an interface and spawn a "gap bridging"-PM. Essentially, the ontology of Figure 1 can one day be reproduced as a server, sub-servers and groups of process managers.

Of course a critical factor is the fact that the KS/PM will never know who is requesting the simulation. Consider the following. Consider the following scenario: an animator at ***Walt Disney Studios*** has created a scene of an animated character pushing a glass of water off a table and into a fire. The stage is set: the glass of water is at the table's edge about to fall into the fire, and the animator must get home to take his daughter to softball practice. Normally, the complete animation of this scene could take days. In this scenario (for which this current NSF project is a test-bed), the animator makes an Internet request to a remote high performance computer to solicit assistance in animating the scene. The remote computer inspects the request and launches the following software programs: dynamics (for the falling glass), finite element codes (for the impact of the glass), computational fluid mechanics (for the spilling and splashing of the water), and heat transfer codes (for the interaction of the water and fire), and, of course, fracture and contact processes. After the processing is completed the digital data describing the completed

scene is delivered back to the animator over the Internet. The animator may not wish such detailed physics, but at least there is an animated sequence and precious time has been saved. This impact of this scenario should not be lost upon the reader: inroads are now being made by animators and computer scientists, yet it is the mechanicians who can bring reality to animations. It is only necessary that mechanicians either develop or learn the necessary fundamentals of the cyber-infrastructure.

This proposed architecture may not be the best. There are additional technologies to exploit and it may even come to pass that large-scale systems such as Globus may be required. However, regardless of what approach emerges, the fundamental premise of this work is that computational mechanicians must learn and embrace first the fundamental technologies – for they are simple and are core to all higher level packages. In this way, computational mechanicians will be able to advance their own view of how to solve coupled problems with an appreciation for the complexity of the physical world. Computational mechanicians are aware of the nuances of mechanics – plasticity, visco-plasticity, damage models and variuos coupled phenomena – and are more in a position to contribute to this emerging field when they have a clearer understanding of some of the core technologies of the cyber-infrastrucutre. It is the goal of the authors of this paper to simply advance some of these core technologies in a context that is accessible to researchers in the field of computational mechanics.

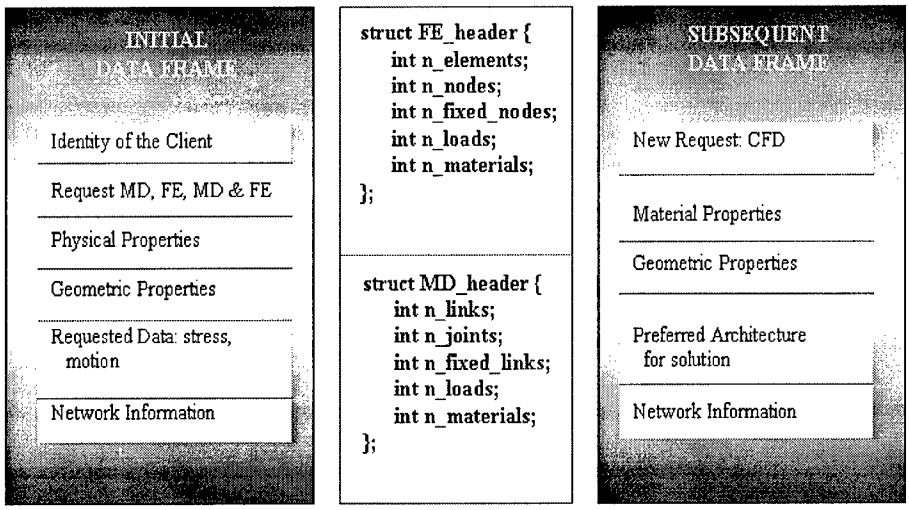

INITIAL DATA FRAME
Identity of the Client
Request MD, FE, MD & FE
Physical Properties
Geometric Properties
Requested Data: stress, motion
Network Information
struct FE_header {
int n_elements;
int n_nodes;
int n_fixed_nodes;
int n_loads;
int n_materials;
};
struct MD_header {
int n_links;
int n_joints;
int n_fixed_links;
int n_loads;
int n_materials;
};
SUBSEQUENT DATA FRAME
New Request: CFD
Material Properties
Geometric Properties
Preferred Architecture for solution
Network Information

FIGURE 1.

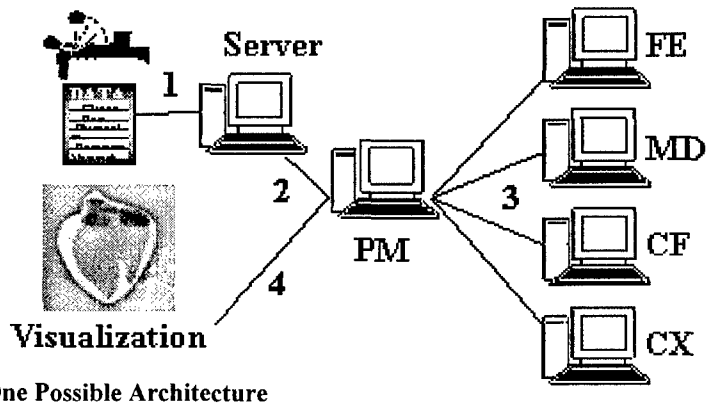


FIGURE 2. **One Possible Architecture**

5. KEY RESEARCH ACCOMPLISHMENTS

1) A paper now in submission to: Journal of Computational Mechanics
Manuscript No. CM-05-0109
Title: A Proposed Cyber-Infrastructure to Solve Multi-Phase Problems in Mechanics
Authors: Impelluso, Thomas

2) Award winning poster:
The work won the first place poster award at the Medicine Meets Virtual Reality 13 conference in Long Beach, January 26.

6. REPORTABLE OUTCOMES

At this time various groups are engaged in solving multi-phase and multi-scale problems. Such solutions are critical for efforts, for example, to endow the digital human with proper physics modeling. The attempts currently being made are using large scale network software to "stitch" together large scale physics codes. While such an approach is in keeping with the traditional scientific method of "standing on the shoulders of those who came before", this approach is actually inimical to the 21st century approach to networked systems.

Traditional legacy physics codes have been built up over many years. Embedded in such codes are "trial and error" algorithms that have managed to work and are now tied directly into the code. These include: older solutions algorithms, constants and variables no longer used, "massage" parameters that 'appeared' to function, early primitive coding styles, first attempts at parallelization, coding that did not 'respect' the CPU (i.e.: cache and page thrashing) and so on.

If one wishes to either endow the digital human with physics, or, even more so, create physics based virtual environments, all these coding styles must be revisited with a new approach to mechanics that respects the need for proper modeling of large deformation and non-linear mechanics. In reality, many of these extant codes simply stitch together solution of simple and traditional problems at the hands of computer scientists.

Future modeling will require a respect for non-linear mechanics.

The outcome of this effort is that the fundamentals of networked software can readily be learned by the computational mechanicians and such individuals can then work with the computer scientists as opposed to simply using the product of computer science efforts.

7. CONCLUSIONS

The algorithms of mechanics – finite elements and multi-body dynamics – can be integrated using the fundamental modules of inter-process communication, without relying on advanced network tools or existing legacy packages.

Such systems can and should be built from the bottom, up, rather than the top-down approach that is currently in vogue.

8. REFERENCES

Atkins, NSF report: http://www.communitytechnology.org/nsf_ci_report/

Belytschko, T. and Mullen, R., Mesh partitions of explicit-implicit time integration, in Formulations and Computational Algorithms in Finite Element Analysis, ed. by J. J. Bathe, J. T. Oden and W. Wunderlich, MIT Press, Cambridge MA 1976, pp. 673-690.

Benson, DJ, 1989, An Efficient, Accurate Simple ALE Method for Nonlinear Finite Element Programs, Computer Methods in Applied Mechanics and Engineering, Vol. 72, 205-350.

Benson, DJ, 1992, Computational Methods in Lagrangian and Eulerian Hydrocodes, *Computer Methods in Applied Mechanics and Engineering*, Vol. 99, pp. 235-394.

Benson, DJ and Hallquist, JO, 1990, A Single Surface Contact Algorithm for the Post-Buckling Analysis of Shell Structures, Comp. Methods Appl. Mech. Eng., 78, 141-163.

Brooks, AN, and Hughes, TJR, 1982, Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations, Computer Methods in Applied Mechanics and Engineering, Vol. 32, 199-259.

Felippa, C.A., and Park, K.C.,Staggered Transient Analysis Procedures for Coupled-Field Mechanical Systems: Formulation, *Computer Methods in Applied Mechanics and Engineering*, Vol. 24, 1980, pp. 61-111.

Felippa, C.A., and Geers, T.L., Partitioned Analysis of Coupled Mechanical Systems, *Engineering Computations*, Vol. 5, 1988, pp. 123-133.

Haug, EJ, 1989, Computer-Aided Kinematics and Dynamics of Mechanical Systems, Volume I: Basic Methods, Allyn and Bacon, Boston.

Hughes, T.J.R., and Liu, W.K., Implicit Explicit Finite Elements in Transient Analysis: I. Stability Theory; II. Implementation and Numerical Examples, *Journal of Applied Mechanics*, Vol. 45, 1978, pp. 371-378.

Impelluso, T. "The Cyber-Infrastructure: A New Research Area for Applied Mechanics", Computer Methods in Visualization. Submitted August 2003.

Impelluso, T. "Physically-based Virtual Reality". GII Testbed and HPC Challenge Applications on the I-WAY, Ed. Holly Koram and M. BrownACM, IEEE, 1995

Impelluso, T. "Distributed, Physically Based-VR with Tactile Feedback." Program Guide ACM SIGGRAPH, Proceedings, August 4-9, 1996. (pp. 55)

Lloyd, D.G., and Besier, T. F., An EMG-driven musculoskeletal model to estimate muscle forces and knee joint movements in vivo. Journal of biomechanics, 36(6), pp. 765-776, 2003

Murakami, H, Benson, DJ, and Takahashi, K, 1992, An Accurate Eulerian Finite Element Code and its Applications to Impact Phenomena, Proceedings of the International Symposium on Impact Engineering, Sendai, Japan, November 2-4, 1992, 79-84.

Murakami, H., Benson, D. J., and Takahashi, K., 1992, An Accurate Eulerian Finite Element Code and its Applications to Impact Phenomena, Proceedings of the International Symposium on Impact Engineering, Sendai, Japan, November 2-4, 1992, 79-84.

Liu, WK, Chang, H, and Belytschko, T, 1988, Arbitrary Lagrangian and Eulerian Petrov-Galerkin Finite Elements for Nonlinear Continua, Computer Methods in Applied Mechanics and Engineering, Vol. 69, 259-310.

Manal, K. and Buchanan, T.S., A one-parameter neural muscle activation model: estimating isometric joint movements from electromyograms. Journal of Biomehcanics, 36(8), pp 1197-2102, 2003

Nikravesh, PE, 1988, Computer-Aided Analysis of Mechanical Systems, Prentice Hall, Englewood Cliffs, NJ.

Noh, WF, and Woodward, Pl, 1976, SLIC (simple line interface calculation), Lecture Notes in Physics 59, Springer Verlag, Heidelberg, 330-340.

Park, K.C., Felippa, C.A.,, Partitioned analysis of coupled systems, Chapter 3 in Computational Methods for Transient Analysis, ed. by T. Belytschko and T. J. R. Hughes, North-Holland, Amsterdam, 1983, pp. 157-219.

Park, K.C., Felippa, C.A., and DeRuntz, J.A., Stabilization of Staggered Solution Procedures for Fluid-Structure Interaction Analysis, in Computational Methods for Fluid-Structure Interaction Problems, ed. by T. Belytschko and T. L. Geers, AMD Vol. 26, American Society of Mechanical Engineers, ASME, New York, 1977, pp. 95-124.

Seguchi, Y, Shindo, A, Tomita, Y and Sunohara, M, 1974, Sliding Rule of Friction in Plastic Forming of Metal, Computational Methods in Nonlinear Mechanics, University of Texas, Austin, 683-692.

van Leer, B, 1979, Towards the Ultimate Conservative Difference Scheme. IV. A Second-Order Sequel to Godunov's Method, Journal of Computational Physics, Vol. 23, 276-299.

Wagner, H, and Blickhan, R. Stabilizing function of antagonistic neuro-musculoskeletal systems: an analytical investigation, Biol Cybern, 89(1), pp 71-79, 2003.